M&G No. 60001.0297US01/305245.01

# REPRESENTING LIST DEFINITIONS AND INSTANCES IN A MARKUP LANGUAGE DOCUMENT

## Related Applications

This patent application is a continuation-in-part application under 35 United States Code § 120 of United States Patent Application No. 10/187,060 filed on June 28, 2002, which is incorporated herein by reference. An exemplary schema in accordance with the present invention is disclosed beginning on page 11 in an application entitled "Mixed Content Flexibility," Serial No. _____, Docket No. 60001.0275US01, filed December 2, 2003, which is hereby incorporated by reference in its entirety.

## Background of the Invention

Markup Languages have attained wide popularity in recent years. One type of markup language, Extensible Markup Language (XML), is a universal language that provides a way to identify, exchange, and process various kinds of data. For example, XML is used to create documents that can be utilized by a variety of application programs. Elements of an XML file have an associated namespace and schema.

In XML, a namespace is a unique identifier for a collection of names that are used in XML documents as element types and attribute names. The name of a namespace is commonly used to uniquely identify each class of XML document. The unique namespaces differentiate markup elements that come from different sources and happen to have the same name.

XML Schemata provide a way to describe and validate data in an XML environment. A schema states what elements and attributes are used to describe content in an XML document, where each element is allowed, what types of text contents are allowed within it and which elements can appear within which other elements. The use of schemata ensures that the document is structured in a consistent manner. Schemata

may be created by a user and generally supported by an associated markup language, such as XML. By using an XML editor, the user can manipulate the XML file and generate XML documents that adhere to the schema the user has created. XML documents may be created to adhere to one or more schemata.

The XML standard is considered by many as the ASCII format of the future, due to its expected pervasiveness throughout the hi-tech industry in the coming years. Recently, some word-processors have begun producing documents that are somewhat XML compatible. For example, some documents may be parsed using an application that understands XML. However, much of the functionality available in word processor documents is not currently available for XML documents.

## Summary of the Invention

The present invention is generally directed towards a method for representing list definitions and instances in a markup language (ML) document such as an XML document. The lists are generated in the ML document as a group of paragraphs, where the paragraph type is one of the list types defined for the document. The lists provide a method for defining a bullet or numbering style as well as providing indentation and format information for a particular list.

More particularly, the present invention relates to representing numbered and bulleted list information in ML so that applications capable of reading a given ML file format, but running in environments where the list generation information has not been installed, are able to still render the lists. The ML document may be manipulated on a server or anywhere even when the application creating the ML document is not present. List definition and instance information (i.e., properties) are saved in a markup language (ML) document without data loss, while allowing the lists to be parsed by ML-aware applications and to be read by ML programmers.

## Brief Description of the Drawings

FIGURE 1 illustrates an exemplary computing device that may be used in one exemplary embodiment of the present invention;

FIGURE 2 is a block diagram illustrating an exemplary environment for practicing the present invention;

FIGURE 3 illustrates an exemplary portion of an ML file that provides a definition for a numbered list that includes multiple levels;

FIGURE 4 illustrates an exemplary portion of an ML file that provides a definition for a picture bulleted list; and

FIGURE 5 shows an exemplary flow diagram for representing list definitions and instances in a ML document, in accordance with aspects of the invention.

## Detailed Description of the Preferred Embodiment

Throughout the specification and claims, the following terms take the meanings explicitly associated herein, unless the context clearly dictates otherwise.

The terms "markup language" or "ML" refer to a language for special codes within a document that specify how parts of the document are to be interpreted by an application. In a word-processor file, the markup language specifies how the text is to be formatted or laid out, whereas in a particular customer schema, the ML tends to specify the text's meaning according to that customer's wishes (e.g., customerName, address, etc). The ML is typically supported by a word-processor and may adhere to the rules of other markup languages, such as XML, while creating further rules of its own.

The term "element" refers to the basic unit of an ML document. The element may contain attributes, other elements, text, and other building blocks for an ML document.

The term "tag" refers to a command inserted in a document that delineates elements within an ML document. Each element can have no more than two tags: the start tag and the end tag. It is possible to have an empty element (with no content) in which case one tag is allowed.

The content between the tags is considered the element's "children" (or descendants). Hence, other elements embedded in the element's content are called

"child elements" or "child nodes" or the element. Text embedded directly in the content of the element is considered the element's "child text nodes". Together, the child elements and the text within an element constitute that element's "content".

The term "attribute" refers to an additional property set to a particular value and associated with the element. Elements may have an arbitrary number of attribute settings associated with them, including none. Attributes are used to associate additional information with an element that will not contain additional elements, or be treated as a text node.

Illustrative Operating Environment

With reference to FIGURE 1, one exemplary system for implementing the invention includes a computing device, such as computing device **100**. In a very basic configuration, computing device **100** typically includes at least one processing unit **102** and system memory **104**. Depending on the exact configuration and type of computing device, system memory **104** may be volatile (such as RAM), non-volatile (such as ROM, flash memory, etc.) or some combination of the two. System memory **104** typically includes an operating system **105**, one or more applications **106**, and may include program data **107**. In one embodiment, application **106** may include a word-processor application **120** that further includes lists **122**. This basic configuration is illustrated in FIGURE 1 by those components within dashed line **108**.

Computing device **100** may have additional features or functionality. For example, computing device **100** may also include additional data storage devices (removable and/or non-removable) such as, for example, magnetic disks, optical disks, or tape. Such additional storage is illustrated in FIGURE 1 by removable storage **109** and non-removable storage **110**. Computer storage media may include volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information, such as computer readable instructions, data structures, program modules, or other data. System memory **104**, removable storage **109** and non-removable storage **110** are all examples of computer storage media. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or

other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by computing device **100**. Any such computer storage media may be part of device **100**. Computing device **100** may also have input device(s) **112** such as keyboard, mouse, pen, voice input device, touch input device, etc. Output device(s) **114** such as a display, speakers, printer, etc. may also be included. These devices are well know in the art and need not be discussed at length here.

Computing device **100** may also contain communication connections **116** that allow the device to communicate with other computing devices **118**, such as over a network. Communication connection **116** is one example of communication media. Communication media may typically be embodied by computer readable instructions, data structures, program modules, or other data in a modulated data signal, such as a carrier wave or other transport mechanism, and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. The term computer readable media as used herein includes both storage media and communication media.

Generally, the present invention is directed at representing numbered and bulleted list definitions and instances in an ML document. The ML document may be read by applications that do not share the same schema that created the document.

FIGURE 2 is a block diagram illustrating an exemplary environment for practicing the present invention. The exemplary environment shown in FIGURE 2 is a word-processor environment **200** that includes word-processor **120**, ML file **210**, ML Schema **215**, and ML validation engine **225**.

In one embodiment, word-processor **120** has its own namespace or namespaces and a schema, or a set of schemas, that is defined for use with documents associated with word-processor **120**. The set of tags and attributes defined by the schema for word-processor **120** define the format of a document to such an extent that it

is referred to as its own native ML. Word-processor **120** internally validates ML file **210**. When validated, the ML elements are examined as to whether they conform to the ML schema **215**. A schema states what tags and attributes are used to describe content in an ML document, where each tag is allowed, and which tags can appear within other tags, ensuring that the documentation is structured the same way. Accordingly, ML **210** is valid when structured as set forth in arbitrary ML schema **215**.

ML validation engine **225** operates similarly to other available validation engines for ML documents. ML validation engine **225** evaluates ML that is in the format of the ML validation engine **225**. For example, XML elements are forwarded to an XML validation engine. In one embodiment, a greater number of validation engines may be associated with word-processor **120** for validating a greater number of ML formats.

Representing List Definitions and Instances in a Markup Language Document

The present invention generally provides a method to represent an application's numbered and bulleted list information in markup language (ML) such as XML. The lists may be parsed by applications that understand the markup other than the application that generated the ML file.

An exemplary list may be the following:

1. One
2. Two

In the list above, there are two items. The items are each members of the same list, and the same list style. A basic representation of the XML structure for the list includes a structure where a list that includes a particular list style is defined at the beginning of the document, and then used in the body of the document to generate the list including the list items. For example, if the list style corresponding to the list above were called "List1", and the Unique ID for that list were "11", then the following document structure may result:

- Document

- Styles
- Lists
  - ListDef
    - ID = 1
  - List
    - val = 1 (This matches to the ID of a list definition)
    - ilfo = 11
- Body
  - Paragraph (StyleName="MsoNormal" ListLevel="1" ListFO="11")
    - TextRun
      - "One"
  - Paragraph (StyleName="MsoNormal" ListLevel="1" ListFO="11")
    - TextRun
      - "Two"

As may be seen, a list item is defined according to a paragraph element that is identified as a member of a particular list defined earlier in the ML document. As is seen in the example structure, the definition of the list occurs at the top of the ML document alongside the definition for the styles used in the document. In one embodiment, the root element of the ML document may contain an element called "<w:lists>" that defines the lists used in the ML document.

Two items of the lists are defined at the top of the document. The first item is the definition of a list type. The definition of the list type identifies all the formatting and other appearance properties for the type of list. Then, for each individual list that exists in the document, a <list> tag is generated after the list definitions. The <list> tag is unique for the list that is referencing it. The <list> tag identifies which list definition it is based on, as well as the tag's ID so that the paragraphs in the document that are part of this list can reference the list properly.

In another embodiment, multi-level lists are supported such that a "lvl" element is included in the lists definition. The lvl tag contains the information for list items that appear on that level. Take the following list for example:

1. One
   a. One A

b. One B
2. Two
a. Two A
i. Two A i

All the items in the above list are substantially the same. However, certain items on the list are on different levels. The lvl tag allows the list to define various elements for each level of the list such as the indentation of each level and the character representing each level. Most of the information for how the list looks may be contained within the various lvl tags. In one embodiment, each lvl tag includes an ilvl attribute which specifies which level is being defined.

Once each list type is defined, the different lists used within the document are defined. List overrides are used when the list items in a document are not part of the same list. An example of multiple lists may include lists such as the following:

1. One
2. Two
3. Three

and:

1. A
2. B
3. C

Even though both lists look the same, they are actually separate lists. If they were the same list, then the "A" would have started at 4 (it would continue the previous one). A list override is generated to reference the same list definitions, but declares itself as being a unique list. The paragraphs in the document that are part of a list then reference the list override.

The following example list includes picture bullets:

- The quick brown fox jumps over the lazy dog.
  - The quick brown fox jumps over the lazy dog.
    - The quick brown fox jumps over the lazy dog.
- The quick brown fox jumps over the lazy dog.

- The quick brown fox jumps over the lazy dog.

To implement the picture bullets in the above list, the image information for the picture bullets is stored within the ML file. In one embodiment, the image information is stored using a "<w:listPicBullet>" element that is a child of the "<w:lists>" element. The listPicBullet element includes a "pict" and an "@listPicBulletId" attribute. The pict element inlcudes the image data for the picture. The listPicBulletId attribute includes the identifier of that picture bullet. List definitions then (i.e., defined by the w:listDef tag) may reference the picture bullet by it's identifier with the <w:lvlPicBulletId> tag.

FIGURE 3 illustrates an exemplary portion of an ML file that provides a definition for a numbered list that includes multiple levels, in accordance with aspects of the present invention.

The definition for the list shown includes two defined levels. It is appreciated that any number of levels may be defined. The list definition provides a variety of information about each list level used within a list, including the list level position on the page, the tab position of the list level, the justification of the list level, a value or identifier of the list level, and other information. The list definition is later used to reference each list level within the body of the ML document.

FIGURE 4 illustrates an exemplary portion of an ML file that provides a definition for a picture bulleted list, in accordance with aspects of the present invention.

The definition of the list shown includes a description of a picture bullet used within a list that is present in the body of the ML file. The picture bullet use is described according to a name that is associated with descriptive binary data for that picture bullet. The list definition also includes an identifier for the picture bullet within the ML file, a style for the picture bullet, a shape identifier and other information that describes the bullet for reference later in the list that is present within the body of the ML file. It is appreciated that any number of picture bullet descriptions may be included within an ML file other than the picture bullet description exemplified in FIGURE 4.

The following is an exemplary portion of schema for generating the list definitions and instances, in accordance with aspects of the present invention:

**Lists Element Defined**

```
<xsd:element name="lists" type="listsElt" minOccurs="0">
    <xsd:annotation>
        <xsd:documentation>Represents the list definitions (for example, the Bullets and Numbering options).</xsd:documentation>
    </xsd:annotation>
</xsd:element>

<xsd:complexType name="listsElt">
    <xsd:annotation>
        <xsd:documentation>Defines a collection of lists.</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
        <xsd:element name="listPicBullet" type="listPicBulletElt" minOccurs="0" maxOccurs="unbounded">
            <xsd:annotation>
                <xsd:documentation>Represents pictures used by picture bullet lists</xsd:documentation>
            </xsd:annotation>
        </xsd:element>
        <xsd:element name="listDef" type="listDefElt" minOccurs="0" maxOccurs="unbounded">
            <xsd:annotation>
                <xsd:documentation>Represents the base list definitions. These definitions will not be directly used in the document. Instead, they will be referenced by list elements.</xsd:documentation>
            </xsd:annotation>
        </xsd:element>
        <xsd:element name="list" type="listElt" minOccurs="0" maxOccurs="unbounded">
            <xsd:annotation>
                <xsd:documentation>Represents the definitions of the lists that are used in the document. The definitions build on top of listDef elements.</xsd:documentation>
            </xsd:annotation>
        </xsd:element>
        <xsd:element name="ilfoMacAtCleanup" type="decimalNumberProperty" minOccurs="0">
            <xsd:annotation>
```

<xsd:documentation>Represents the optional index of the last list when a list cleanup was attempted. This element is used to prevent trying to clean up the lists again.</xsd:documentation>
</xsd:annotation>
</xsd:element>
</xsd:sequence>
</xsd:complexType>

**List Picture Bullet Element**

```
<xsd:complexType name="listPicBulletElt">
    <xsd:annotation>
        <xsd:documentation>Defines the picture used for a list bullet.</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
        <xsd:element name="pict" type="pictureType">
            <xsd:annotation>
                <xsd:documentation>Represents the picture data for the picture bullet.</xsd:documentation>
            </xsd:annotation>
        </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="listPicBulletId" type="decimalNumberType" use="required">
        <xsd:annotation>
            <xsd:documentation>Gets or sets the picture bullet ID</xsd:documentation>
        </xsd:annotation>
    </xsd:attribute>
</xsd:complexType>

<xsd:complexType name="pictureType">
    <xsd:annotation>
        <xsd:documentation>Defines a picture.</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence maxOccurs="unbounded">
        <xsd:element name="binData" type="binDataType" minOccurs="0">
            <xsd:annotation>
                <xsd:documentation>Contains the binary data representing this object.</xsd:documentation>
            </xsd:annotation>
        </xsd:element>
        <xsd:element name="movie" type="binDataType" minOccurs="0">
```

```
                    <xsd:annotation>
                        <xsd:documentation>Contains the binary data
representing this movie.</xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
                <xsd:choice minOccurs="0" maxOccurs="1">
                    <xsd:element name="background" minOccurs="0"
type="backgroundElt">
                        <xsd:annotation>
                            <xsd:documentation>Represents the background
for this document.</xsd:documentation>
                        </xsd:annotation>
                    </xsd:element>
                    <xsd:element name="applet" type="appletElt">
                        <xsd:annotation>
                            <xsd:documentation>Contains data for the Java
applet this object represents</xsd:documentation>
                        </xsd:annotation>
                    </xsd:element>
                    <xsd:element name="msAccessHTML" type="stringType">
                        <xsd:annotation>
                            <xsd:documentation>Contains data for the Access
Web Bot this object represents.</xsd:documentation>
                        </xsd:annotation>
                    </xsd:element>
                    <xsd:element name="scriptAnchor" type="scriptAnchorType">
                        <xsd:annotation>
                            <xsd:documentation>Contains data for the script
anchor this object represents.</xsd:documentation>
                        </xsd:annotation>
                    </xsd:element>
                    <xsd:element name="ocx" type="ocxType" minOccurs="0">
                        <xsd:annotation>
                            <xsd:documentation>Contains data for the OCX
control this object represents.</xsd:documentation>
                        </xsd:annotation>
                    </xsd:element>
                </xsd:choice>
                <xsd:any processContents="skip" namespace="urn:schemas-microsoft-
com:vml" minOccurs="0"></xsd:any>
                <xsd:any processContents="skip" namespace="urn:schemas-microsoft-
com:office:office" minOccurs="0"></xsd:any>
            </xsd:sequence>
</xsd:complexType>
```

**List Definition Element (This is used to define a type of list)**

```
<xsd:complexType name="listDefElt">
    <xsd:annotation>
        <xsd:documentation>Defines the base list definitions. These definitions
5   will not be directly used in the document. Instead, they will be referenced by list
elements.</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
        <xsd:element name="lsid" type="longHexNumberProperty"
10  minOccurs="0">
            <xsd:annotation>
                <xsd:documentation>Represents the list
ID.</xsd:documentation>
            </xsd:annotation>
15      </xsd:element>
        <xsd:element name="plt" type="listTypeProperty" minOccurs="0">
            <xsd:annotation>
                <xsd:documentation>Specifies the list
type.</xsd:documentation>
20          </xsd:annotation>
        </xsd:element>
        <xsd:element name="tmpl" type="longHexNumberProperty"
minOccurs="0">
            <xsd:annotation>
25              <xsd:documentation>Specifies the list template used for
formatting the list.</xsd:documentation>
            </xsd:annotation>
        </xsd:element>
        <xsd:element name="name" type="stringProperty" minOccurs="0">
30          <xsd:annotation>
                <xsd:documentation>Represents the list's
name.</xsd:documentation>
            </xsd:annotation>
        </xsd:element>
35      <xsd:element name="styleLink" type="stringProperty" minOccurs="0">
            <xsd:annotation>
                <xsd:documentation>Represents the name of the list style
defined by this list.</xsd:documentation>
            </xsd:annotation>
40      </xsd:element>
        <xsd:element name="listStyleLink" type="stringProperty"
minOccurs="0">
            <xsd:annotation>
```

```
                    <xsd:documentation>Represents the name of the list style
that the list is referencing.</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="lvl" type="lvlElt" minOccurs="0"
maxOccurs="9">
                <xsd:annotation>
                    <xsd:documentation>Defines a level in this
list.</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="listDefId" type="decimalNumberType" use="required">
            <xsd:annotation>
                <xsd:documentation>Gets or sets the identifier of this list
definition.</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
</xsd:complexType>

<xsd:complexType name="listTypeProperty">
    <xsd:annotation>
        <xsd:documentation>Defines the type of list (single, multi,
hybrid).</xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="val" type="listTypeValues" use="required">
        <xsd:annotation>
            <xsd:documentation>Defines the type of list (single, multi, or
hybrid).</xsd:documentation>
        </xsd:annotation>
    </xsd:attribute>
</xsd:complexType>

<xsd:simpleType name="listTypeValues">
    <xsd:annotation>
        <xsd:documentation>Defines the type of list (single, multi, or
hybrid).</xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="SingleLevel"></xsd:enumeration>
        <xsd:enumeration value="Multilevel"></xsd:enumeration>
        <xsd:enumeration value="HybridMultilevel"></xsd:enumeration>
    </xsd:restriction>
```

```
</xsd:simpleType>


<xsd:complexType name="lvlElt">
    <xsd:annotation>
        <xsd:documentation>Defines a list level.</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
        <xsd:element name="start" type="decimalNumberProperty"
minOccurs="0">
            <xsd:annotation>
                <xsd:documentation>Represents the starting number for
this list.</xsd:documentation>
            </xsd:annotation>
        </xsd:element>
        <xsd:element name="nfc" type="decimalNumberProperty"
minOccurs="0">
            <xsd:annotation>
                <xsd:documentation>Specifies the number style used for
a list.</xsd:documentation>
            </xsd:annotation>
        </xsd:element>
        <xsd:element name="lvlRestart" type="decimalNumberProperty"
minOccurs="0">
            <xsd:annotation>
                <xsd:documentation>Represents the number at which to
restart the list numbering.</xsd:documentation>
            </xsd:annotation>
        </xsd:element>
        <xsd:element name="pStyle" type="stringProperty" minOccurs="0">
            <xsd:annotation>
                <xsd:documentation>Represents the paragraph style for
the list numbers.</xsd:documentation>
            </xsd:annotation>
        </xsd:element>
        <xsd:element name="isLgl" type="onOffProperty" minOccurs="0">
            <xsd:annotation>
                <xsd:documentation>Specifies whether this level is
following the legal numbering rules.</xsd:documentation>
            </xsd:annotation>
        </xsd:element>
        <xsd:element name="suff" type="listLevelSuffixProperty"
minOccurs="0">
            <xsd:annotation>
```

```
                        <xsd:documentation>Specifies what follows the list
number.</xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="lvlText" type="listLevelTextProperty"
minOccurs="0">
                    <xsd:annotation>
                        <xsd:documentation>Represents the number format
text.</xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="lvlPicBulletId" type="decimalNumberProperty"
minOccurs="0">
                    <xsd:annotation>
                        <xsd:documentation>Represents the picture bullet
ID.</xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="legacy" type="lvlLegacyElt" minOccurs="0">
                    <xsd:annotation>
                        <xsd:documentation>Specifies whether the list level is
from Word 6.0/95 or earlier.</xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="lvlJc" type="jcProperty" minOccurs="0">
                    <xsd:annotation>
                        <xsd:documentation>Represents justification of the actual
number</xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="pPr" type="pPrElt" minOccurs="0">
                    <xsd:annotation>
                        <xsd:documentation>Represents the paragraph
properties</xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="rPr" type="rPrElt" minOccurs="0">
                    <xsd:annotation>
                        <xsd:documentation>Represents the run properties for the
list numbers.</xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="ilvl" type="decimalNumberType" use="required">
                <xsd:annotation>
```

```
                    <xsd:documentation>Gets or sets the level
number.</xsd:documentation>
                </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="tplc" type="longHexNumberType" use="optional">
                <xsd:annotation>
                    <xsd:documentation>Gets or sets the template code, which is
used to figure out which gallery item to display in bullets and
numbering.</xsd:documentation>
                </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="tentative" type="onOffType" use="optional">
                <xsd:annotation>
                    <xsd:documentation>Specifies whether the level's format is still
tentative.</xsd:documentation>
                </xsd:annotation>
            </xsd:attribute>
        </xsd:complexType>


        <xsd:complexType name="listLevelSuffixProperty">
            <xsd:annotation>
                <xsd:documentation>Defines what follows the list number (for example,
a tab, a space, or nothing).</xsd:documentation>
            </xsd:annotation>
            <xsd:attribute name="val" type="listLevelSuffixValues" use="required">
                <xsd:annotation>
                    <xsd:documentation>Specifies what follows the list
number.</xsd:documentation>
                </xsd:annotation>
            </xsd:attribute>
        </xsd:complexType>


        <xsd:simpleType name="listLevelSuffixValues">
            <xsd:annotation>
                <xsd:documentation>Specifies what follows the list number (for
example, a tab, a space, or nothing).</xsd:documentation>
            </xsd:annotation>
            <xsd:restriction base="xsd:string">
                <xsd:enumeration value="Tab"></xsd:enumeration>
                <xsd:enumeration value="Space"></xsd:enumeration>
                <xsd:enumeration value="Nothing"></xsd:enumeration>
            </xsd:restriction>
        </xsd:simpleType>
```

```
<xsd:complexType name="listLevelTextProperty">
        <xsd:annotation>
                <xsd:documentation>Specifies the text for the
level.</xsd:documentation>
        </xsd:annotation>
        <xsd:attribute name="val" type="stringType" use="optional">
                <xsd:annotation>
                        <xsd:documentation>Gets or sets the number format
text.</xsd:documentation>
                </xsd:annotation>
        </xsd:attribute>
        <xsd:attribute name="null" type="onOffType" use="optional">
                <xsd:annotation>
                        <xsd:documentation>Specifies whether the string is null. A null
string is different from an empty string. If the level text is a string with only a null
character, set the null attribute to on.</xsd:documentation>
                </xsd:annotation>
        </xsd:attribute>
</xsd:complexType>


<xsd:complexType name="lvlLegacyElt">
        <xsd:annotation>
                <xsd:documentation>Specifies that the list level is from Word 6.0/95 or
earlier.</xsd:documentation>
        </xsd:annotation>
        <xsd:attribute name="legacy" type="onOffType" use="optional">
                <xsd:annotation>
                        <xsd:documentation>Specifies whether this list level is from
Word 6.0/95 or earlier. If set to on, the list level is from Word 6.0/95 or
earlier.</xsd:documentation>
                </xsd:annotation>
        </xsd:attribute>
        <xsd:attribute name="legacySpace" type="twipsMeasureType" use="optional">
                <xsd:annotation>
                        <xsd:documentation>Specifies how much space to insert between
the number and the text (for Word 6.0/95 or earlier).</xsd:documentation>
                </xsd:annotation>
        </xsd:attribute>
        <xsd:attribute name="legacyIndent" type="signedTwipsMeasureType"
use="optional">
                <xsd:annotation>
```

```
                    <xsd:documentation>Specifies how much space to indent the
number by (for Word 6.0/95 or earlier).</xsd:documentation>
                </xsd:annotation>
            </xsd:attribute>
</xsd:complexType>


<xsd:complexType name="jcProperty">
        <xsd:annotation>
            <xsd:documentation>Defines a property that uses a justification
setting.</xsd:documentation>
        </xsd:annotation>
        <xsd:attribute name="val" type="jcValue" use="required">
            <xsd:annotation>
                <xsd:documentation>Gets or sets the value of a justification
setting.</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
</xsd:complexType>


<xsd:simpleType name="jcValue">
        <xsd:annotation>
            <xsd:documentation>Defines a justification
setting.</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="left"></xsd:enumeration>
            <xsd:enumeration value="center"></xsd:enumeration>
            <xsd:enumeration value="right"></xsd:enumeration>
            <xsd:enumeration value="both">
                <xsd:annotation>
                    <xsd:documentation>Both left and right
justified.</xsd:documentation>
                </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration value="medium-kashida"></xsd:enumeration>
            <xsd:enumeration value="distribute"></xsd:enumeration>
            <xsd:enumeration value="list-tab"></xsd:enumeration>
            <xsd:enumeration value="high-kashida"></xsd:enumeration>
            <xsd:enumeration value="low-kashida"></xsd:enumeration>
            <xsd:enumeration value="thai-distribute"></xsd:enumeration>
        </xsd:restriction>
</xsd:simpleType>
```

**List Element (This defines the list instance)**

```
<xsd:complexType name="listElt">
    <xsd:annotation>
        <xsd:documentation>Defines a list.</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
        <xsd:element name="ilst" type="decimalNumberProperty"
minOccurs="1">
            <xsd:annotation>
                <xsd:documentation>Represents the identifier of which
list definition this list uses (not the lsid element).</xsd:documentation>
            </xsd:annotation>
        </xsd:element>
        <xsd:element name="lvlOverride" type="listLvlElt" minOccurs="0"
maxOccurs="9">
            <xsd:annotation>
                <xsd:documentation>Defines a list level that overrides a
previously defined list level.</xsd:documentation>
            </xsd:annotation>
        </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="ilfo" type="decimalNumberType" use="required">
        <xsd:annotation>
            <xsd:documentation>Gets or sets the identifier of which list this
is in the document.</xsd:documentation>
        </xsd:annotation>
    </xsd:attribute>
</xsd:complexType>


<xsd:complexType name="listLvlElt">
    <xsd:annotation>
        <xsd:documentation>Defines the settings for the list
level.</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
        <xsd:element name="startOverride" type="decimalNumberProperty"
minOccurs="0">
            <xsd:annotation>
                <xsd:documentation>Represents the number this level
starts at (overrides the list starting number).</xsd:documentation>
            </xsd:annotation>
        </xsd:element>
```

```
                <xsd:element name="lvl" type="lvlElt" minOccurs="0"
maxOccurs="1">
                        <xsd:annotation>
                                <xsd:documentation>Defines a list
level.</xsd:documentation>
                        </xsd:annotation>
                </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="ilvl" type="decimalNumberType" use="required">
                <xsd:annotation>
                        <xsd:documentation>Gets or sets the list level
number.</xsd:documentation>
                </xsd:annotation>
        </xsd:attribute>
</xsd:complexType>
```

FIGURE 5 shows an exemplary flow diagram for representing list definitions and instances in a ML document, in accordance with aspects of the invention. After start block **510**, the process flows to block **520** where the list information within a document such as a word-processor document, is determined. The list information used within a document may include many different types of lists such as bulleted lists and numbered lists, including those that are not natively supported by later applications parsing the document. Once the list information is determined, processing proceeds to decision block **530**.

At decision block **530**, a determination is made whether the document includes a list that corresponds to a picture bulleted list. When the list being examined is not a picture bulleted list, processing advances to block **550**. However, if the list is a picture bulleted list, processing moves to block **540**.

At block **540**, the "pict" element and the "listPicBulletID" attribute are included among the elements, attributes, and values to which the list properties will be mapped. The "pict" element includes the image data for the picture that comprises the bullet, and the "listPicBulletID" attribute includes the id of that picture bullet. After the "pict" element and the "listPicBulletID" attribute are included, processing continues at decision block **550**.

At decision block **550**, a determination is made whether the list being examined is a new list within a document that already includes a list. If the list is not a

new list, but is rather the first list within the document, processing advances to block **570**. However, if the list is new list within a document that already includes a list, processing continues at block **560**.

At block **560**, a list override is included for mapping the list properties so that the multiple lists are separated in the ML file. In one embodiment, a list override is necessary with consecutive lists within the document that are not separated by other text. In other embodiments, a list override is included between each list of the document despite intervening fields within the document. Including the list override ensures that the correct bullet or number corresponds to the correct list item when the ML file is parsed. Processing continues at block **570**.

At block **570**, the properties of the lists within the document are mapped into elements, attributes, and values of the ML file. The lists and the properties associated with the lists may change from page to page, section to section, chapter to chapter and the like. There may be more than one mapping, therefore, per document. Once the list properties are mapped, or written to the ML file, processing moves to block **580**.

At block **580**, the properties of the lists are stored in a ML document that may be read by applications that understand the ML. Once the properties are stored, processing moves to end block **590** and returns to processing other actions.

The above specification, examples and data provide a complete description of the manufacture and use of the composition of the invention. Since many embodiments of the invention can be made without departing from the spirit and scope of the invention, the invention resides in the claims hereinafter appended.